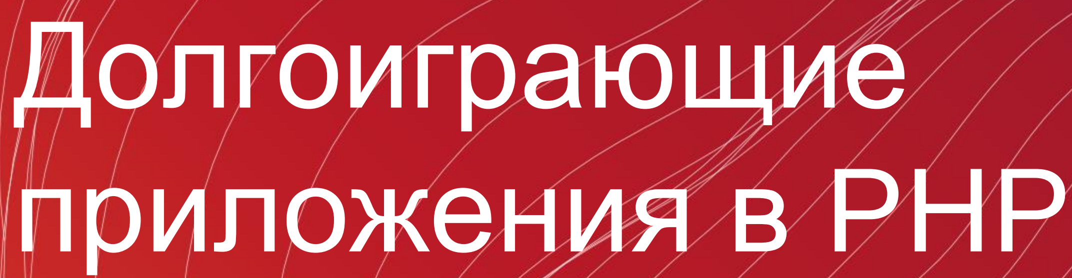


Долгоиграющие приложения в PHP

Александр Пряхин
Авито



Александр Пряхин

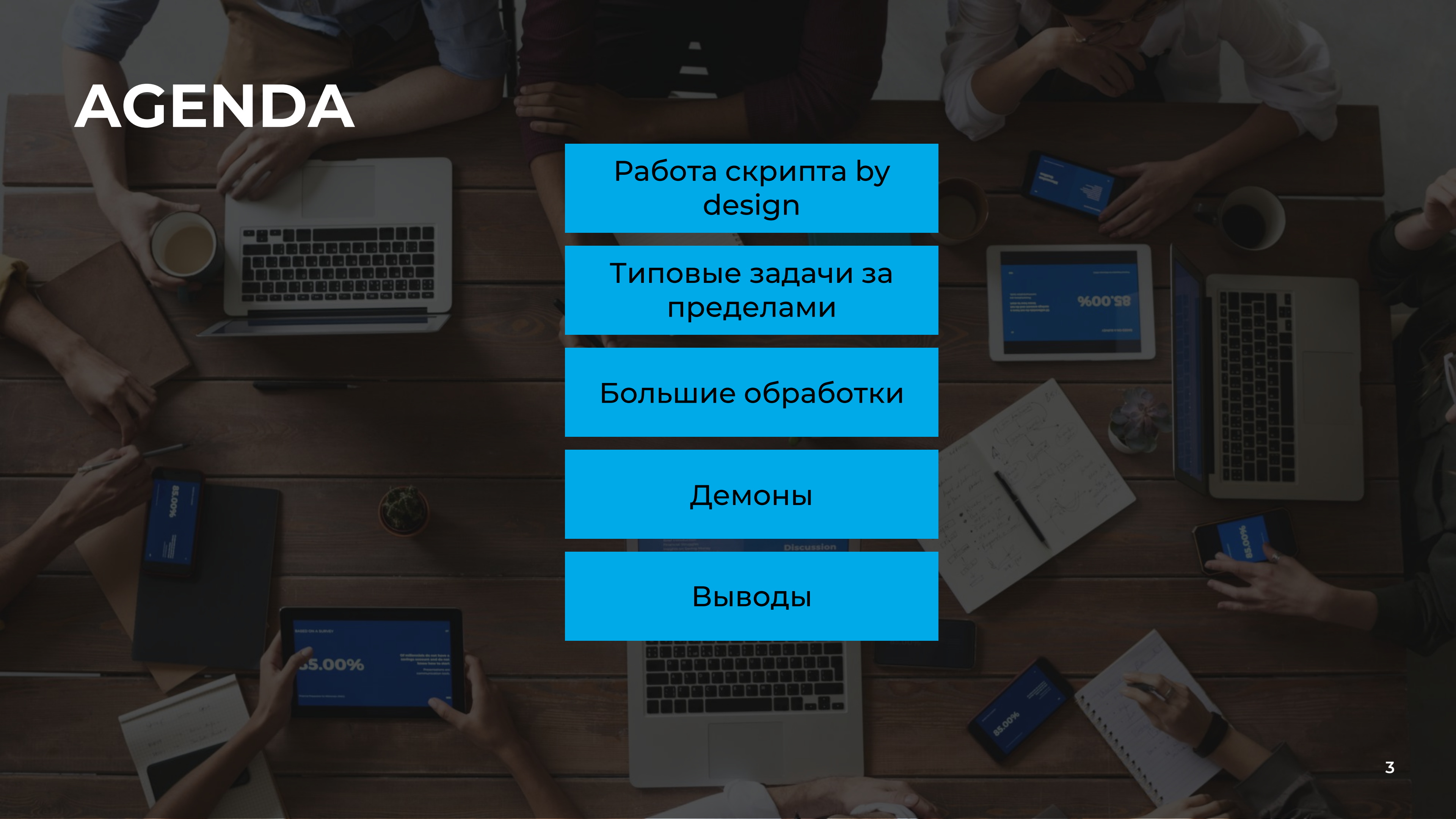
Technical Unit Lead

14+ лет в IT
8+ лет руковожу командами
Преподаю и менторю

▶ avpryakhin@avito.ru



AGENDA



- Работа скрипта by design
- Типовые задачи за пределами
- Большие обработки
- Демоны
- Выводы

Рождён, чтобы умирать

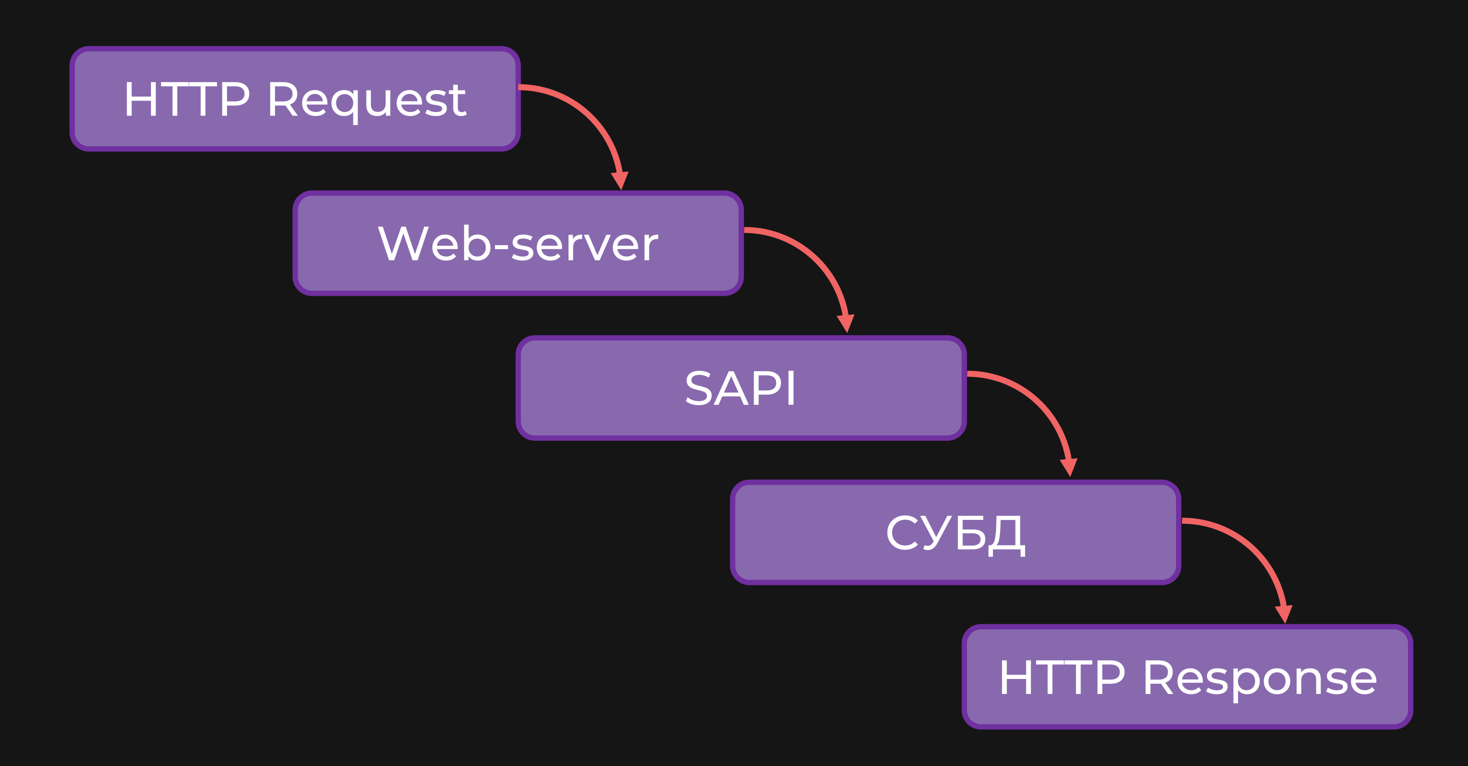


LONG TIME AGO IN A GALAXY FAR AWAY…

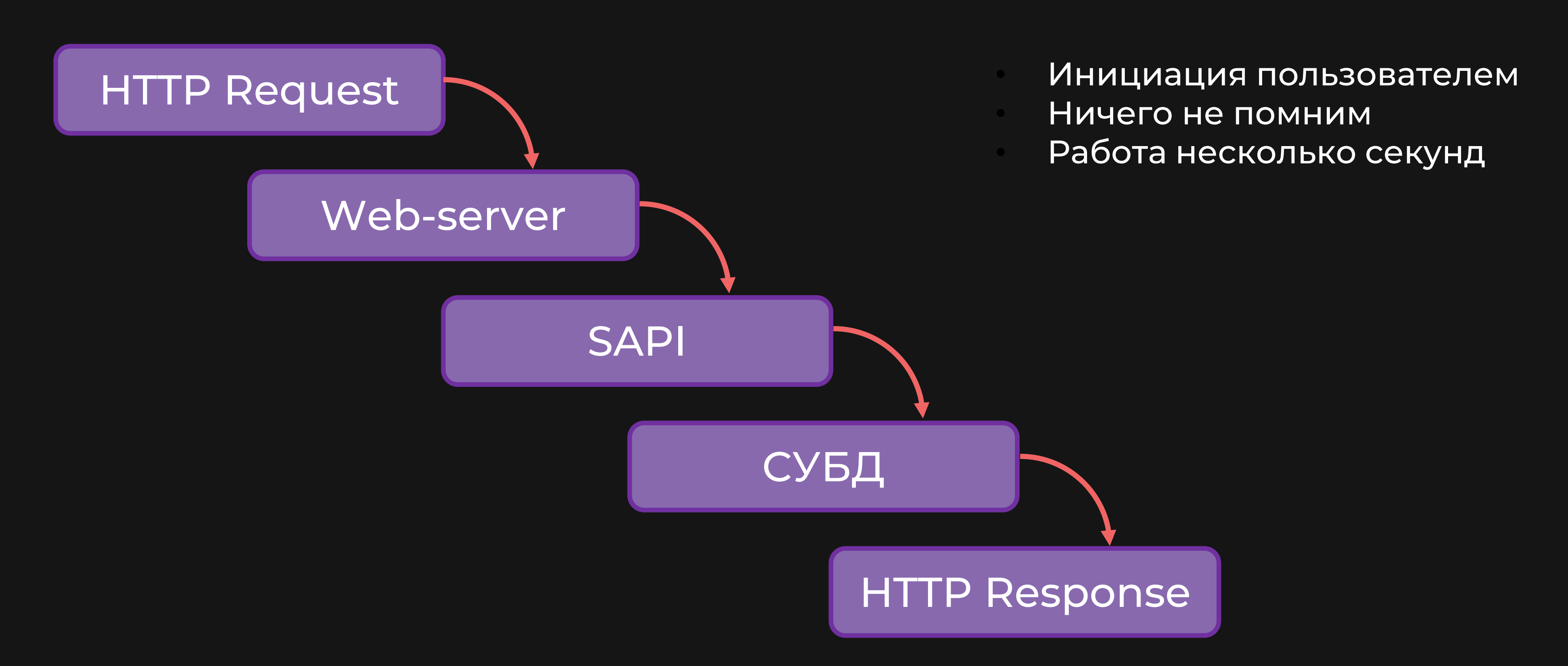
- PHP плохо обрабатывает большие данные
- PHP имеет утечки памяти
- PHP не умеет в многопоточность
- И вообще, есть же Go!



PHP BY DESIGN



PHP BY DESIGN



- Инициация пользователем
- Ничего не помним
- Работа несколько секунд

ПОТРЕБНОСТИ

Обработки не раз в минуту, а по event'ам

Хочется отчёт за год

Слушать очередь событий

А ещё можно отчёт со срезом по когортам?

ETL -> DWH

КЛАСТЕРЫ ПРОБЛЕМ

Обработка большого объёма данных

Данных немного, но надо реагировать сразу

Всё это работает дольше нескольких секунд

Есть же другие языки!



РОСТ СТЕКА

Команда учит новый язык

РОСТ СТЕКА

Команда учит новый язык

- Нет экспертизы на старте
- Команда дорожает

РОСТ СТЕКА

Команда учит новый язык

- Нет экспертизы на старте
- Команда дорожает

Нанимаем нового специалиста

РОСТ СТЕКА

Команда учит новый язык

- Нет экспертизы на старте
- Команда дорожает

Нанимаем нового специалиста

- Будет ли достаточно задач?
- Явное удорожание

РОСТ СТЕКА

Команда учит новый язык

- Нет экспертизы на старте
- Команда дорожает

Нанимаем нового специалиста

- Будет ли достаточно задач?
- Явное удорожание

А так ли плох PHP?

Большие обработки



ЧТО ХОТИМ СДЕЛАТЬ?

Построить отчет

ЧТО ХОТИМ СДЕЛАТЬ?

Построить отчет

Набираем сырые данные

ЧТО ХОТИМ СДЕЛАТЬ?

Построить отчет

Набираем сырые данные

Трансформируем

ЧТО ХОТИМ СДЕЛАТЬ?

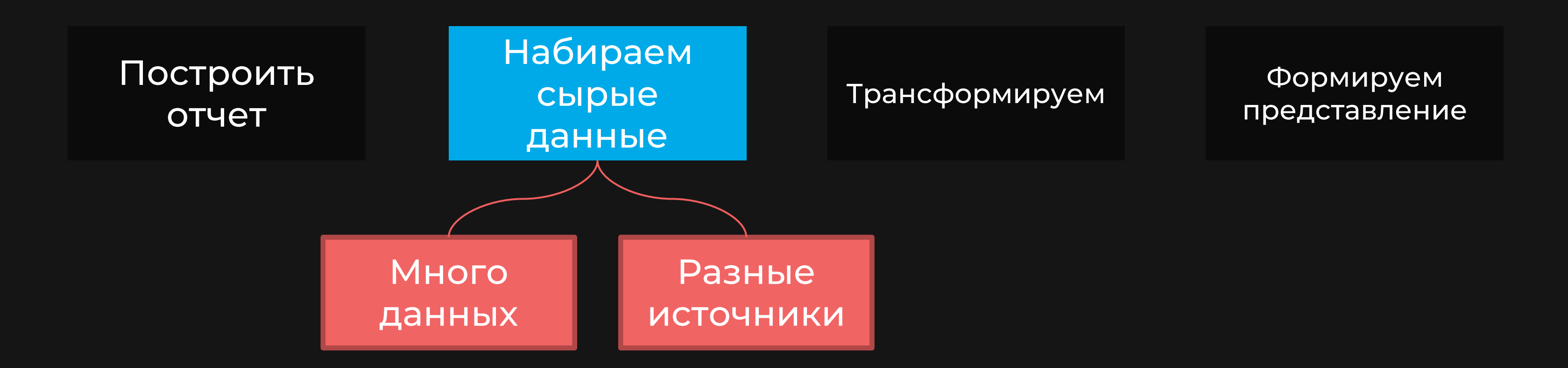
Построить отчет

Набираем сырые данные

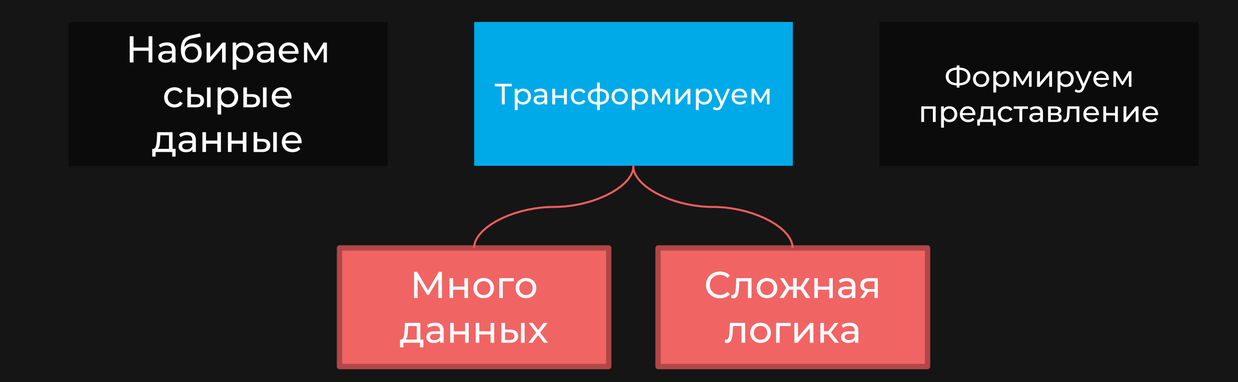
Трансформируем

Формируем представление

С ЧЕМ МОЖЕМ СТОЛКНУТЬСЯ



С ЧЕМ МОЖЕМ СТОЛКНУТЬСЯ



РАСХОД ПАМЯТИ

Правило: потоковая обработка □ читаем только то, что хотим обработать здесь и сейчас

Примеры:

- Получение данных порциями из большого файла

Типовая ошибка: «Я хочу получить всё»

РАСХОД ПАМЯТИ

Правило: потоковая обработка - читаем только то, что хотим обработать здесь и сейчас

Примеры:

- Чтение большого файла/потока с данными

```
function readTheFile(string $path): \Generator {
    $handle = fopen($path, "r");

    while(!feof($handle)) {
        yield trim(fgets($handle));
    }

    fclose($handle);
}
```

РАСХОД ПАМЯТИ

```
function readTheFile(string $path): \Generator {
    $handle = fopen($path, "r");

    while(!feof($handle)) {
        yield trim(fgets($handle));
    }

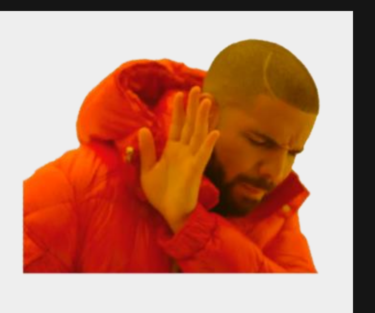
    fclose($handle);
}
```

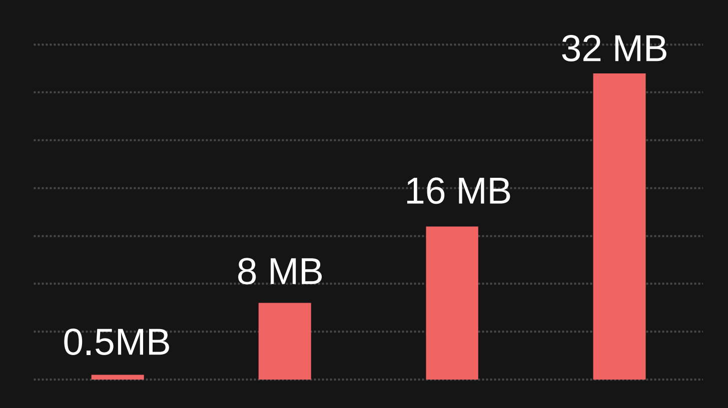
- Объем памяти равен размеру самой большой итерируемой части
- Не происходит аллоцирование памяти для промежуточного хранения
- На выходе также нужно соблюдать правила обработки результата!

РАСХОД ПАМЯТИ

```
$generator = readTheFile($path);
$result = null;

foreach($generator as $item) {
    $result .= $item;
}
```

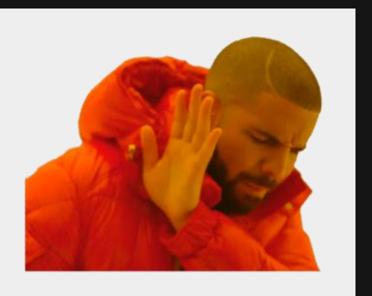




РАСХОД ПАМЯТИ

```php
$generator = readTheFile($path);
$result = null;

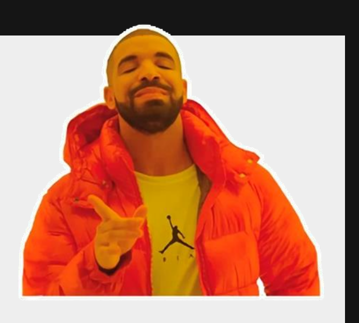
foreach($generator as $item) {
    $result .= $item;
}
```



32 MB
16 MB
8 MB
0.5MB

```php
$generator = readTheFile($path);
$result = null;

foreach($generator as $item) {
    processItem($item);
}
```





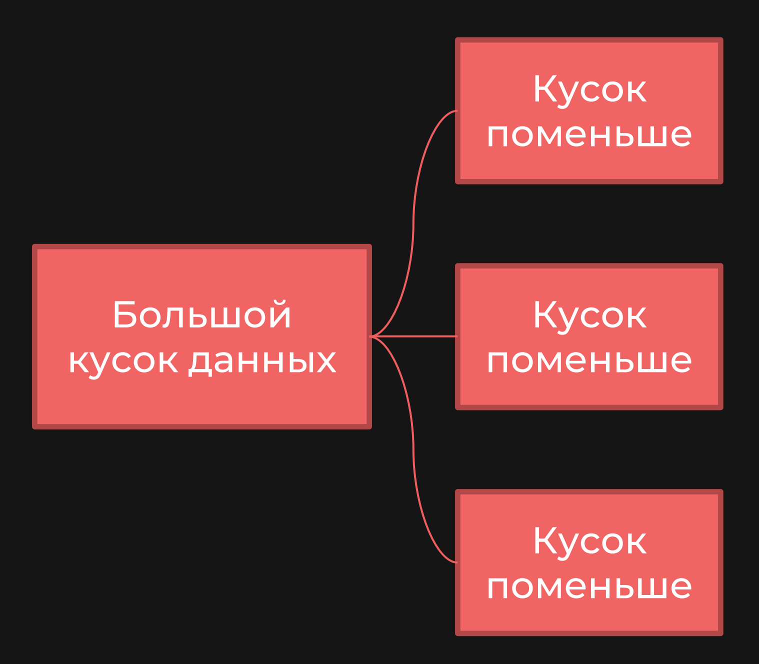
НО ЭТО ДОЛГО!



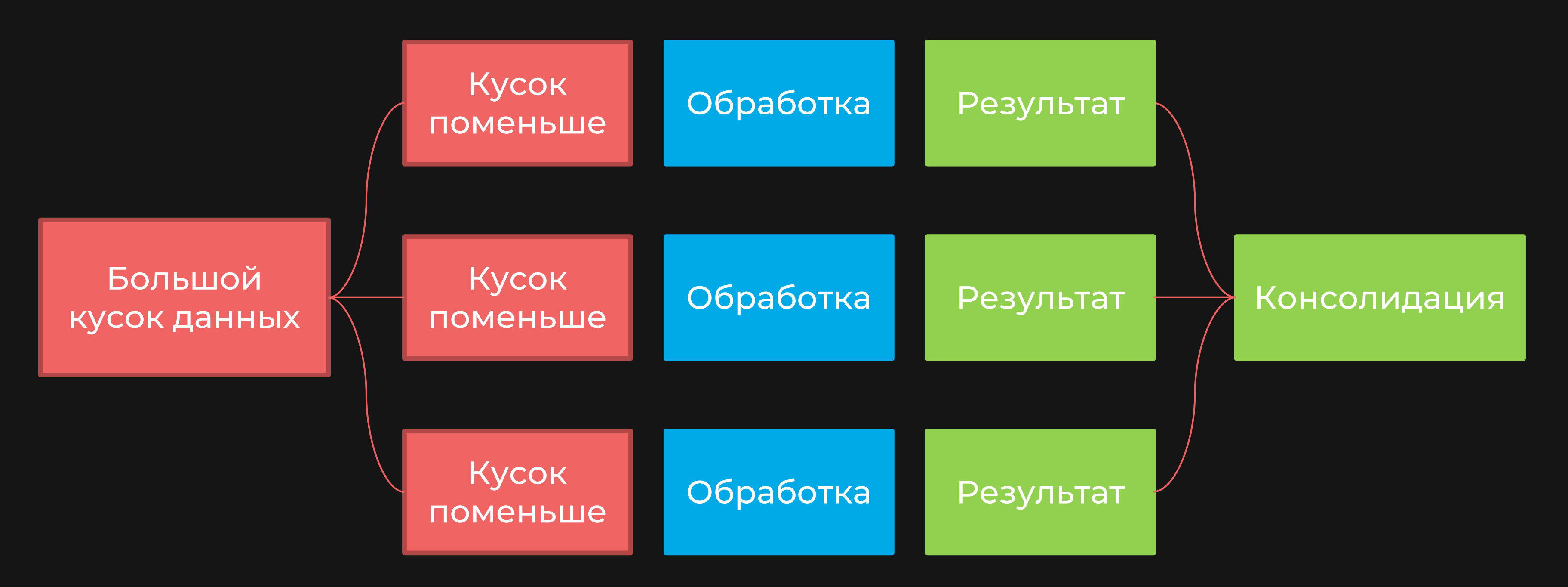
ДАВАЙТЕ РАСПАРАЛЛЕЛИМ!

Большой
кусок данных

ДАВАЙТЕ РАСПАРАЛЛЕЛИМ!



ДАВАЙТЕ РАСПАРАЛЛЕЛИМ!



ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В PHP

Swoole

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В PHP

Swoole

- Асинхронное выполнение
- Переключение контекстов
- Кэш между воркерами
- Много чего ещё



Пар…

Swoole

- Аси…
- Пер…
- Кэш…
- Мно…

Ещё один доклад про RoadRunner, AmPHP и Swoole =)

См. перевод

Нравится · 1 | Ответить · 1 ответ

Alexander Pryakhin Автор

1 мес. •••

Tech Unit Lead @ Avito

Ну не совсем) Я больше хочу зайти в организацию самого процесса, какие проблемы там можно встретить. Фреймворк - это уже надстройка, решающая (или нет) эти проблемы. А вот в суть хотелось бы погрузиться

Нравится · 1 | Ответить

ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В PHP

Parallel

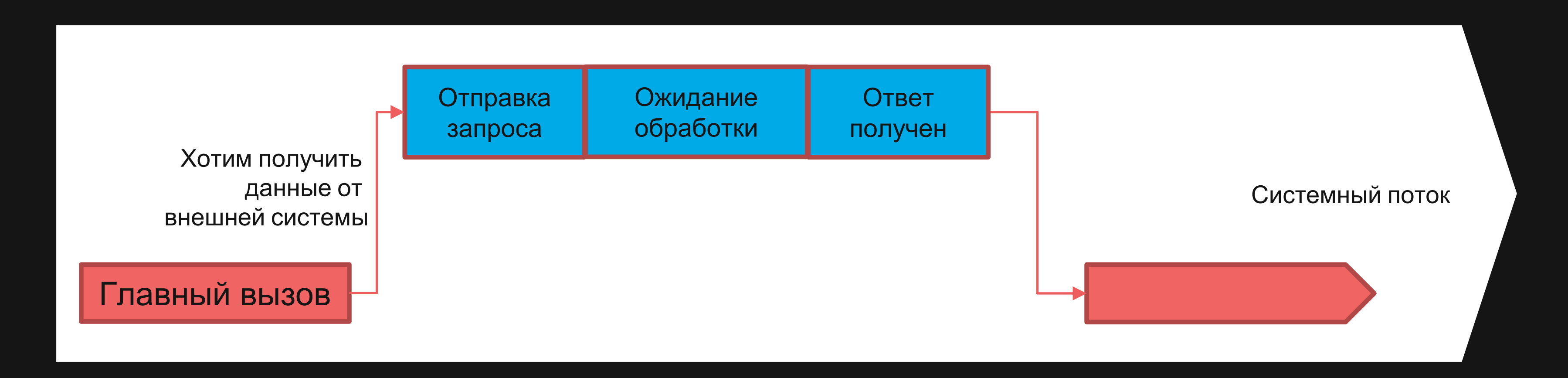
ПАРАЛЛЕЛЬНЫЕ ВЫЧИСЛЕНИЯ В PHP

Parallel

- Замена pthreads
- ZTS
- 7.2+
- Потоки, а не корутины

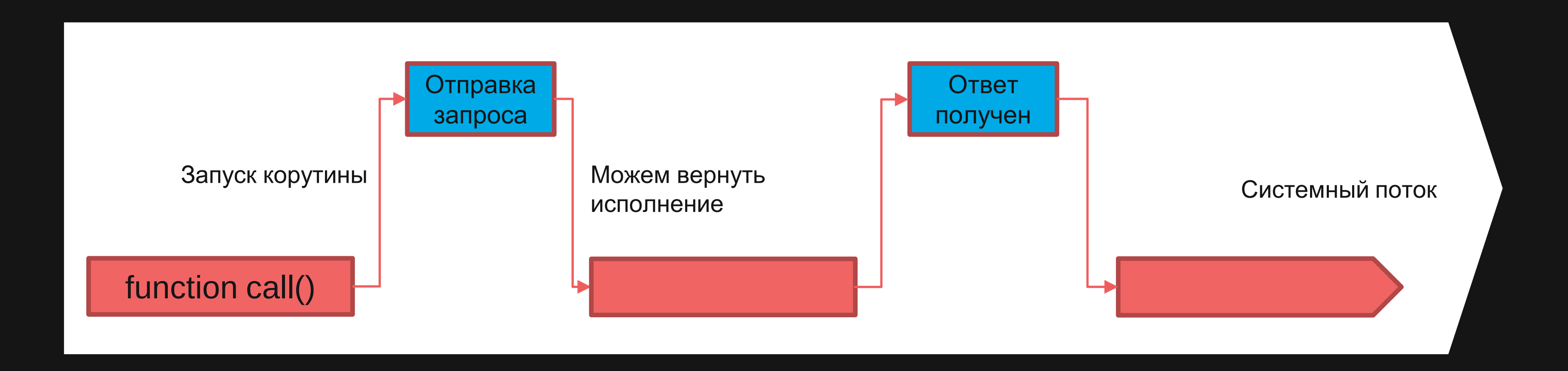


СИНХРОННОЕ ВЫПОЛНЕНИЕ КОДА



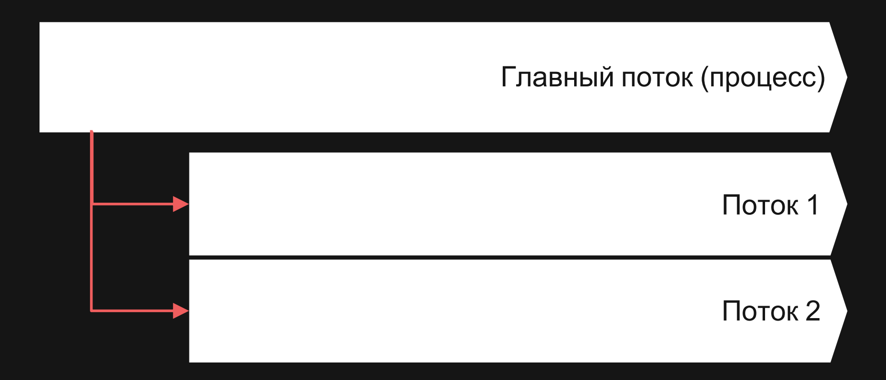
Один поток выполнения
Стек вызова функций
Вызывающая функция ждёт полного завершения вызванной

КОРУТИНЫ



Не требуются переключения на системном уровне
Не требуют поддержки со стороны ОС
Асинхронны (события происходят независимо от главного потока)

ПОТОКИ



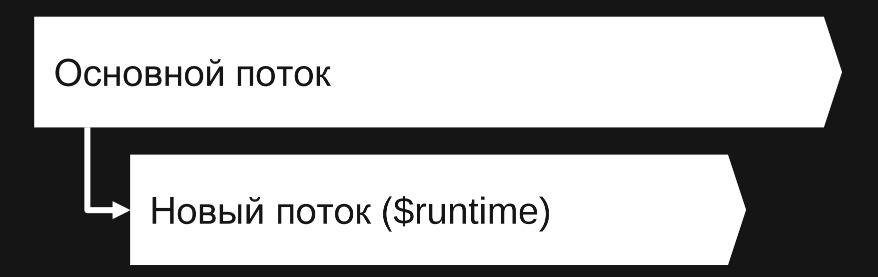
Потоки независимы
Создание потока – более дорогая операция
Выполнение параллельно

COROUTINE VS THREADS

	Coroutine	Threads
Суть	Не блокирует основной поток, но работают внутри Асинхронная работа	Создают выделенные потоки Параллельная работа
Специализация	Снижение нагрузки на систему	Ускорение сложных вычислений

КАК ЭТО РАБОТАЕТ: ЗАПУСК ПОТОКА

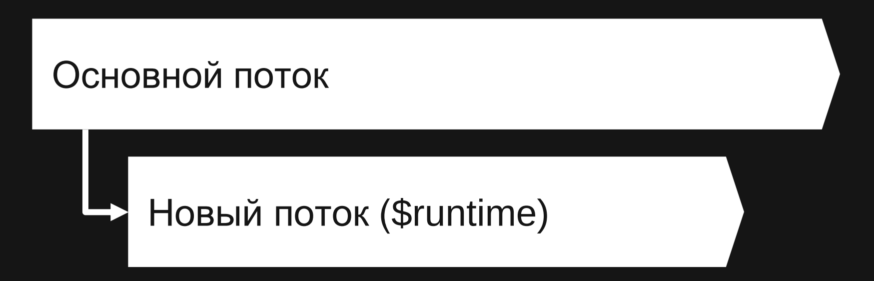
```
$runtime = new \parallel\Runtime();
$runtime->run($task);
```



КАК ЭТО РАБОТАЕТ: ЗАМЫКАНИЯ

```
/** @var Closure */
$task = function() { /* Что-то делаем
*/ }

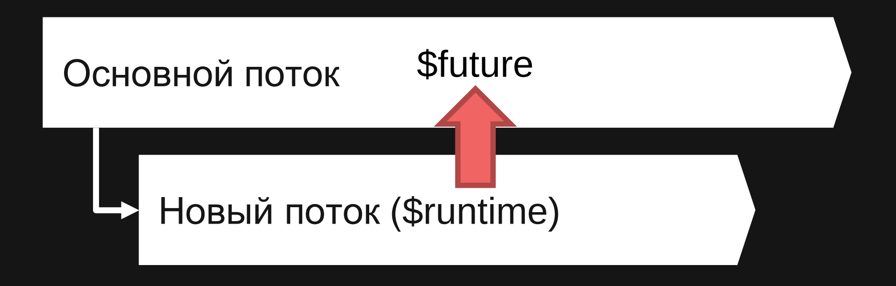
$runtime = new \parallel\Runtime();
$runtime->run($task);
```



КАК ЭТО РАБОТАЕТ: FUTURE

```
/** @var Closure */
$task = function() { /* Что-то делаем
*/ }

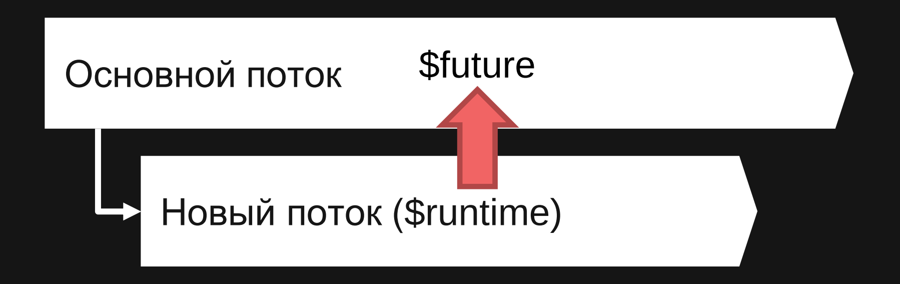
$runtime = new \parallel\Runtime();
$future = $runtime->run($task);
```



КАК ЭТО РАБОТАЕТ: FUTURE

```
/** @var Closure */
$task = function() {
    /* Что-то делаем */
    return $counter;
}

$runtime = new \parallel\Runtime();
$future = $runtime->run($task);
echo $future->value();
```

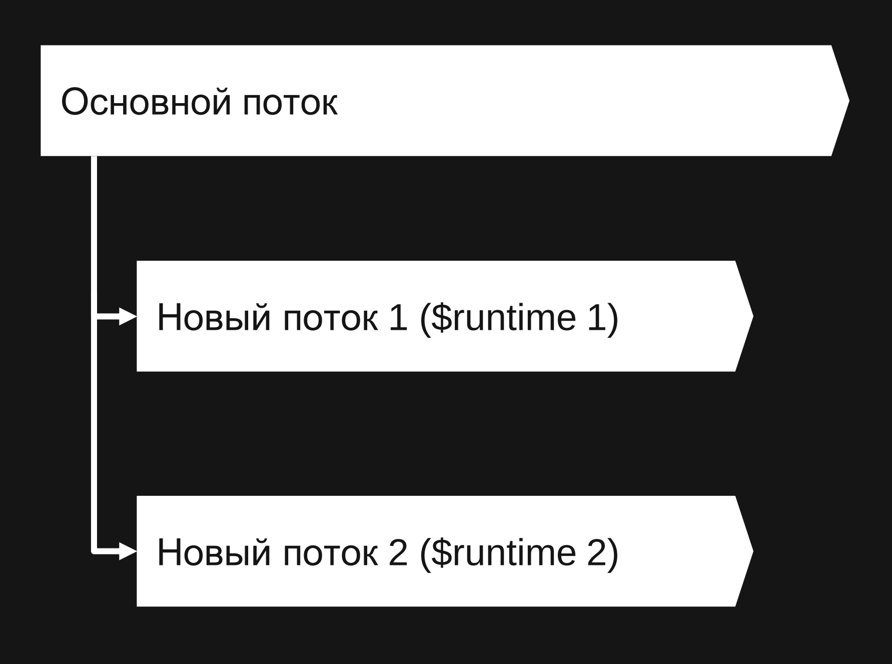


avito.tech

КАК ЭТО РАБОТАЕТ: ШАРИНГ

```
/** @var Closure */
$task = function() { /* Что-то делаем
*/ }

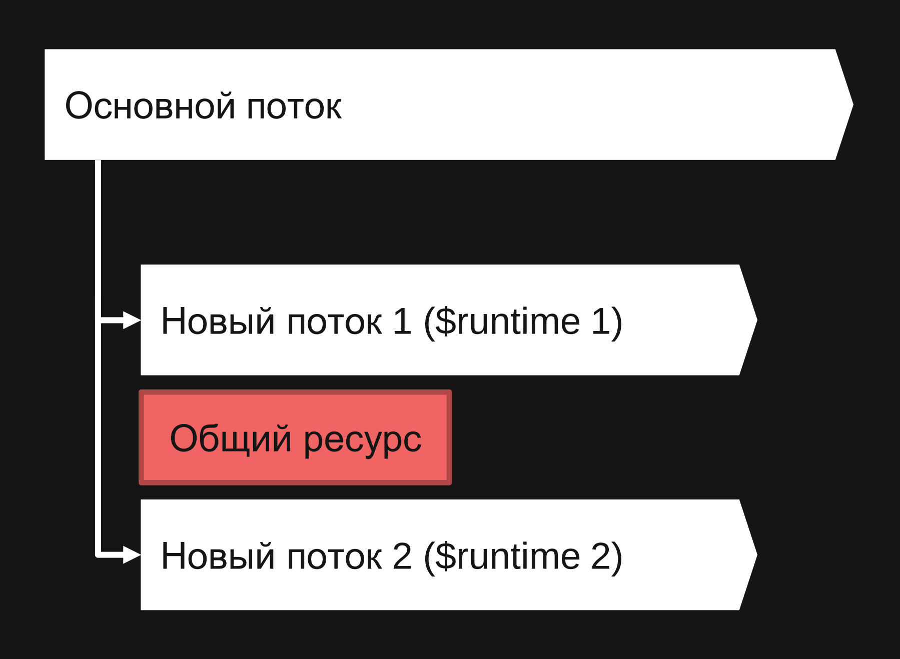
$runtime1 = new \parallel\Runtime();
$runtime2 = new \parallel\Runtime();
```



КАК ЭТО РАБОТАЕТ: ШАРИНГ

```
/** @var Closure */
$task = function() { /* Что-то делаем
*/ }

$runtime1 = new \parallel\Runtime();
$runtime2 = new \parallel\Runtime();
```



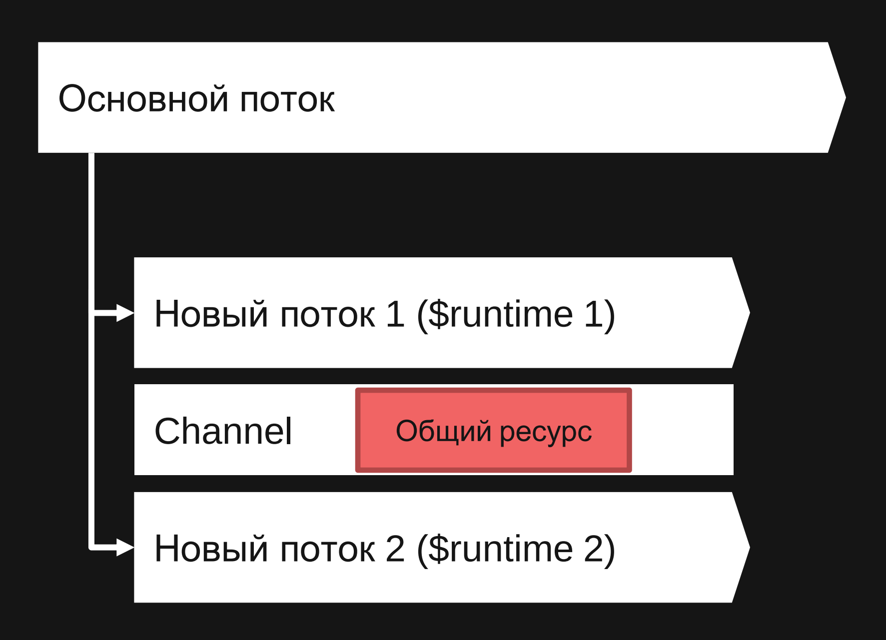
КАК ЭТО РАБОТАЕТ: ШАРИНГ

```
$channel = new \parallel\Channel();

/** @var Closure */
$task = function(Channel $ch) { /*
Что-то делаем */ }

$runtime1 = new \parallel\Runtime();
$runtime2 = new \parallel\Runtime();

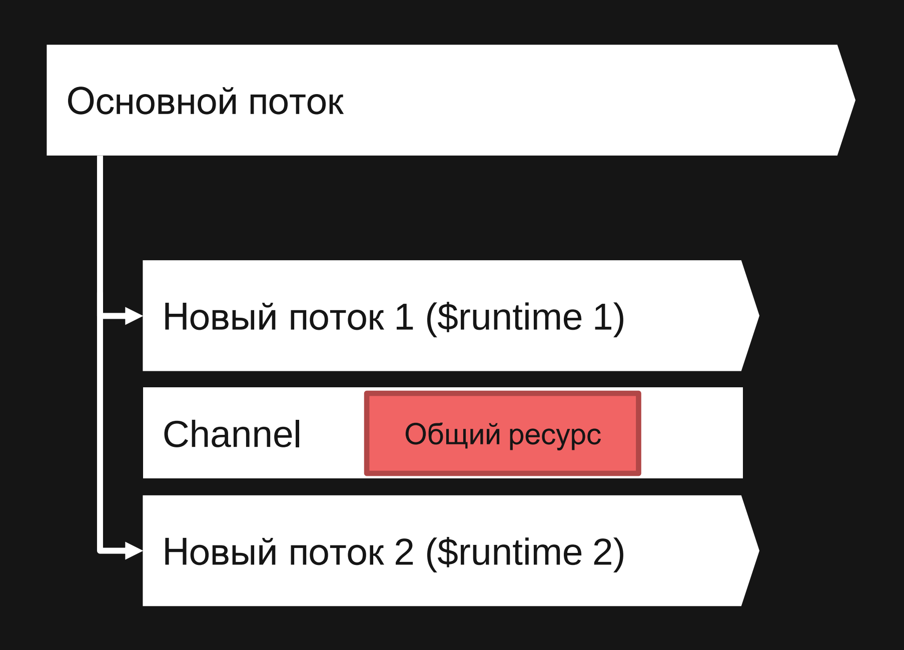
$runtime1->run($task);
$runtime2->run($task);
```



КАК ЭТО РАБОТАЕТ: ШАРИНГ

```
/** @var Closure */
$task = function(Channel $ch) {
     /* Что-то делаем */

     $value = $ch->recv(); //
получаем данные
     $ch->send(++$value); //
отправляем данные
}
```



ЧТО ЕЩЕ УМЕЕТ PARALLEL

- Поддерживается передача замыканий в каналы
- Можно писать свои мьютексы (класс Sync)
- Можно работать с event loop (класс Event)

КАК ЭТО РАБОТАЕТ

Бенчмарк с примерами и комментариями



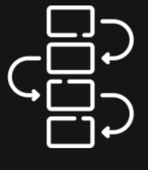
СПЕЦИФИКА PARALLEL



Очень скудная документация и небольшой опыт сообщества



Сборка в Docker заставит потеть

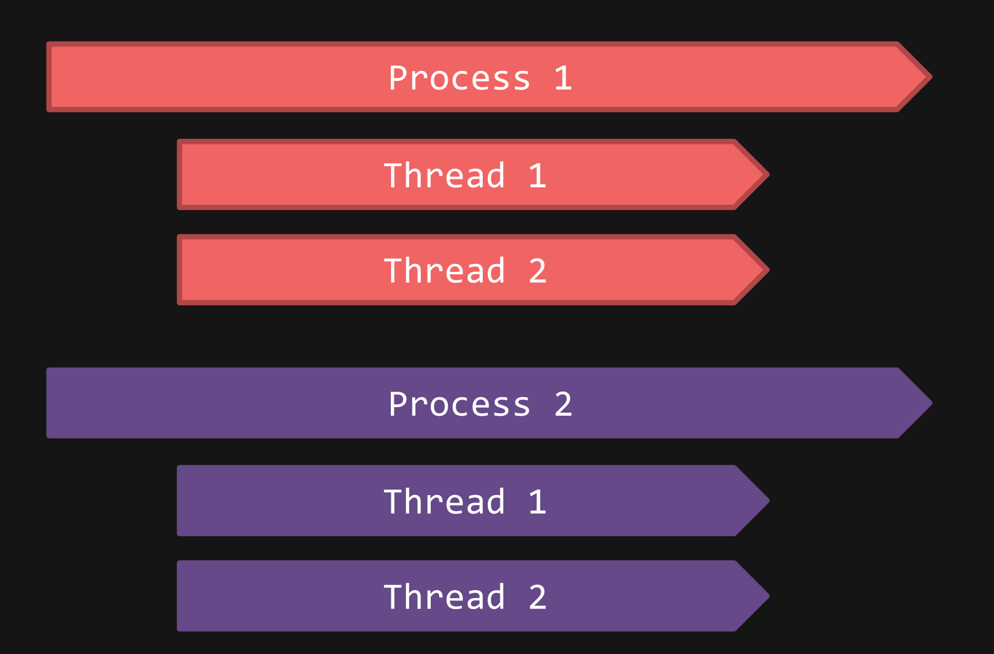


Дебаг потоков специфичен и требует адаптации



Не снизит расход ресурсов

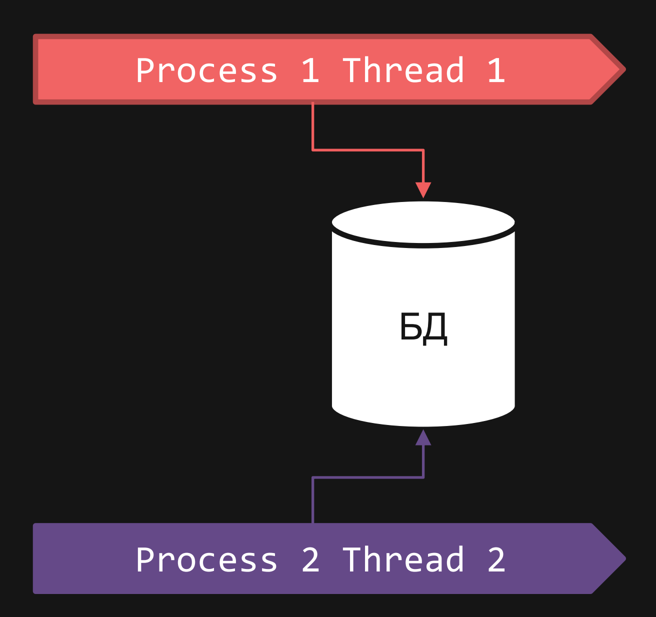
ШАРИНГ РЕСУРСОВ



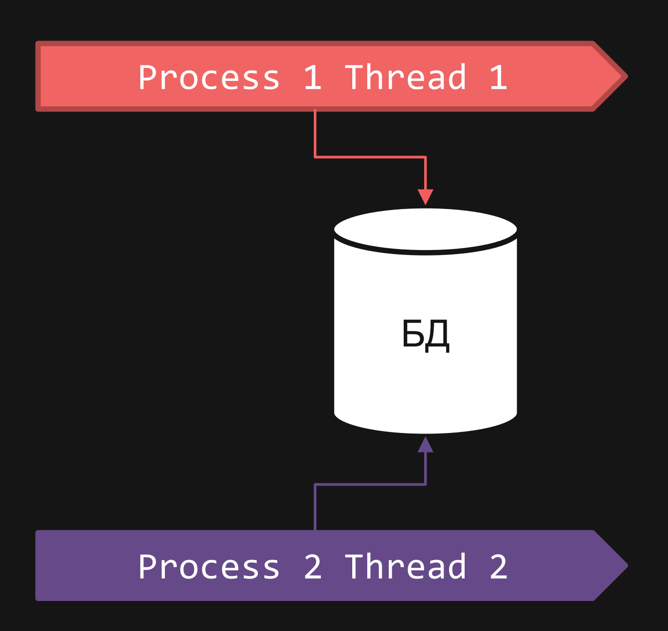
Шаринг ресурсов возможен только между потоками.

Между процессами обмен только через внешние системы.

ШАРИНГ РЕСУРСОВ

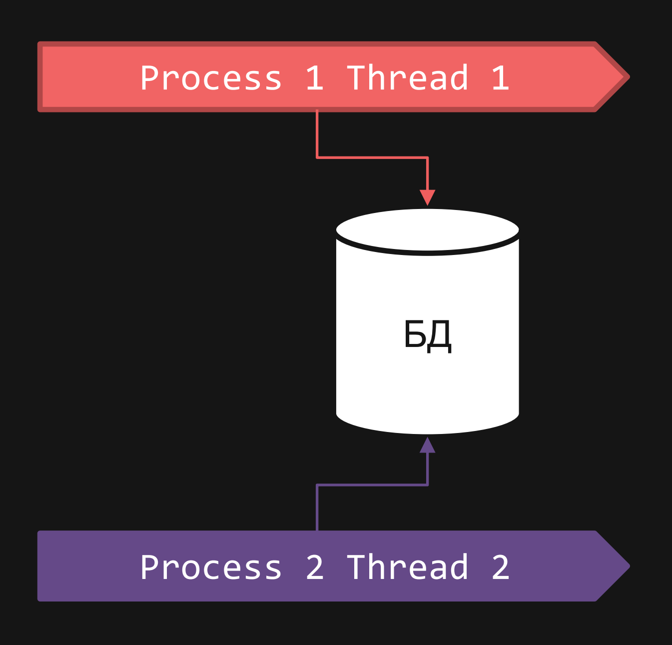


ШАРИНГ РЕСУРСОВ

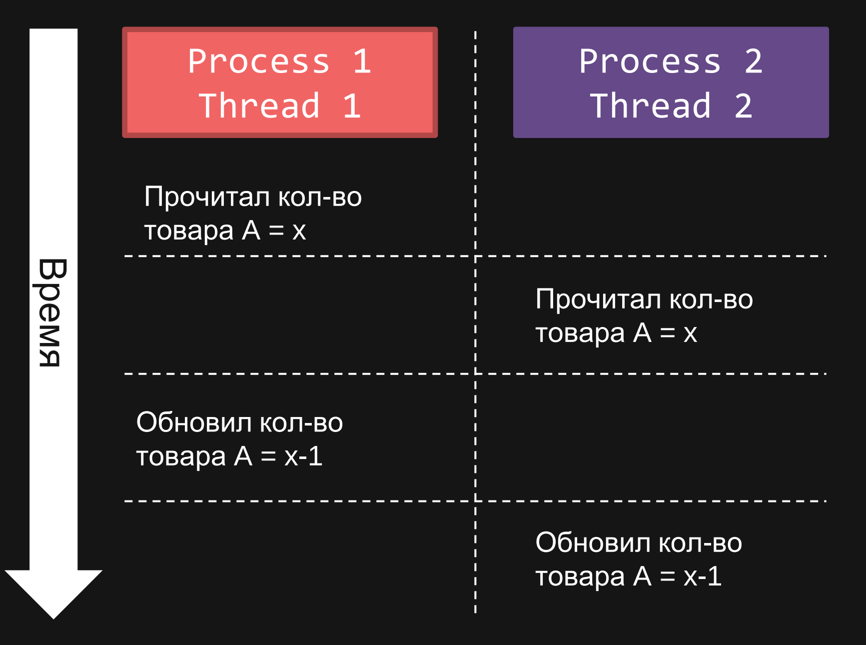


Состояние гонки

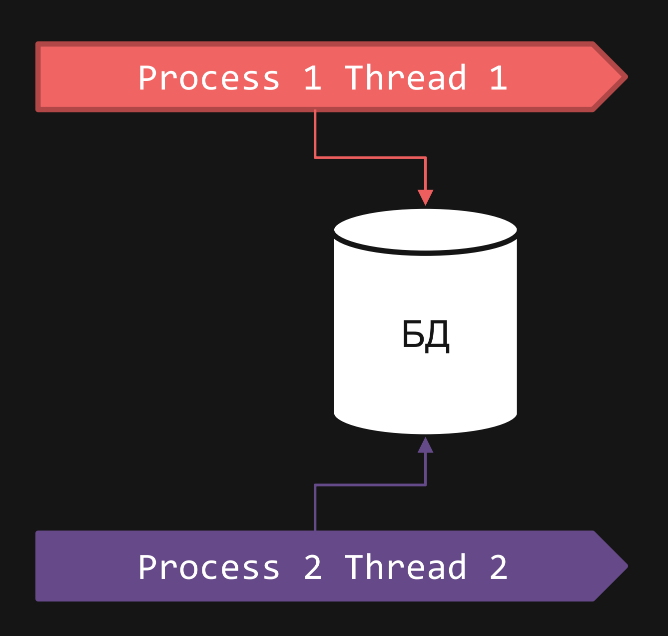
ШАРИНГ РЕСУРСОВ



Состояние гонки



ШАРИНГ РЕСУРСОВ

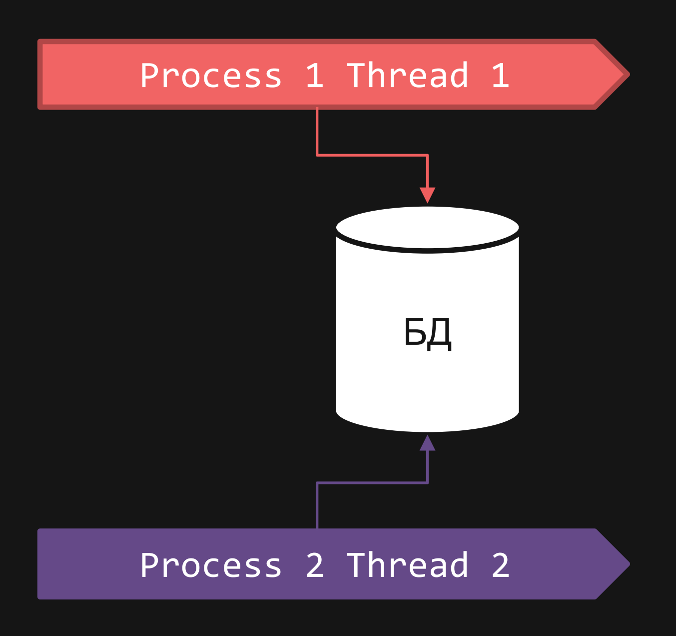


Состояние гонки



Вводим транзакции

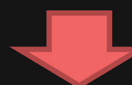
ШАРИНГ РЕСУРСОВ



Состояние гонки

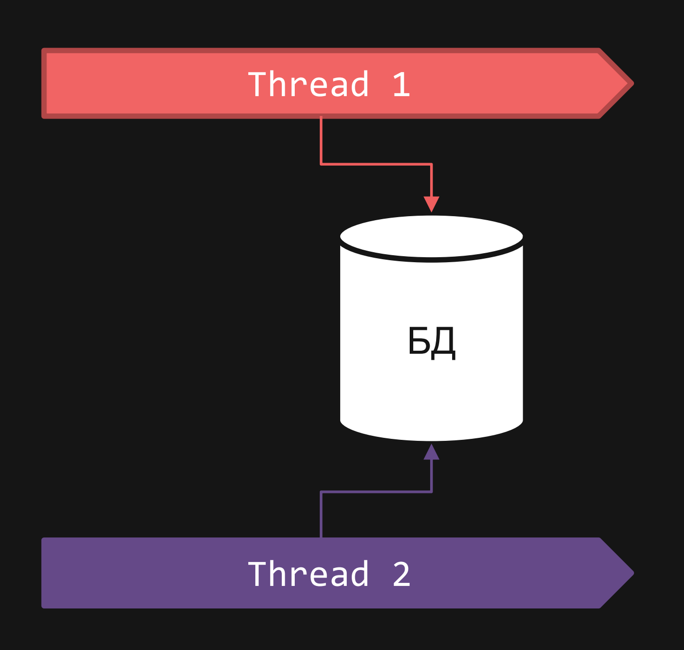


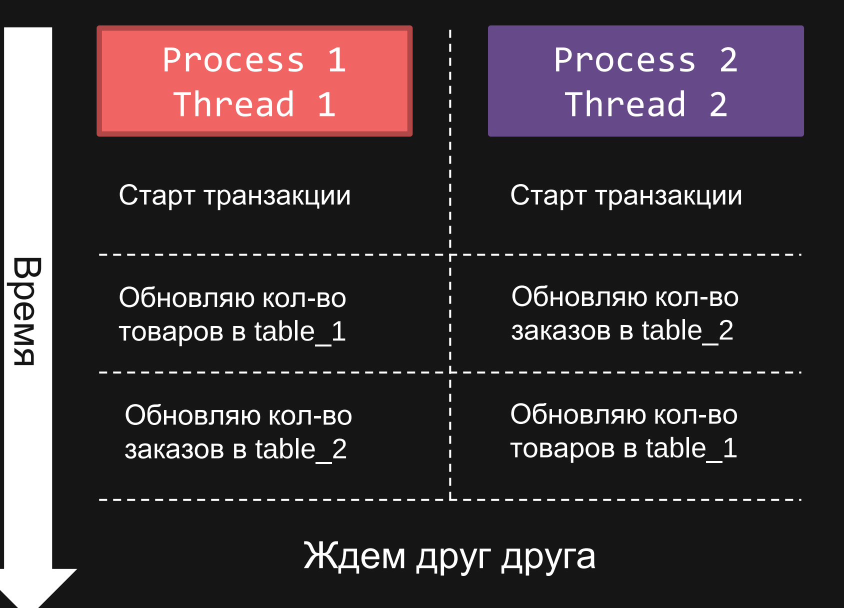
Вводим транзакции



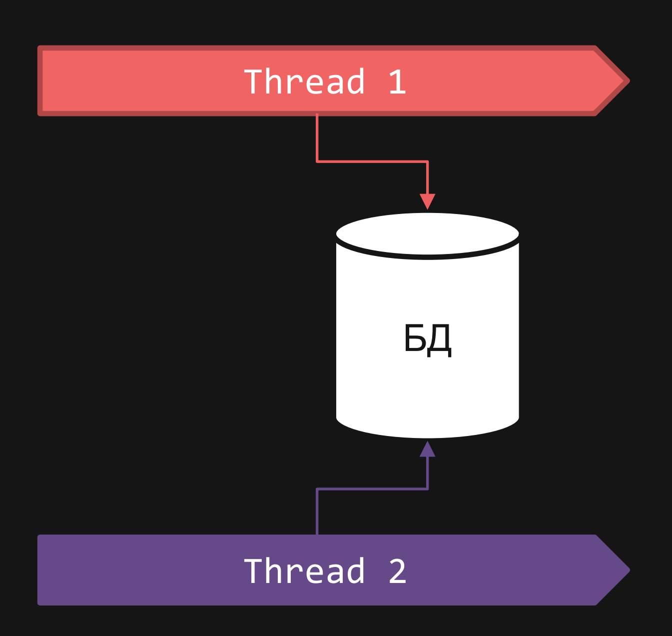
Снижаем скорость работы

ШАРИНГ РЕСУРСОВ





ШАРИНГ РЕСУРСОВ



Deadlock

- Атомарность транзакций

ШАРИНГ РЕСУРСОВ

Session 1:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 1;
```

Session 2:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 2;
UPDATE accounts SET balance = balance -
100.00 WHERE acctnum = 1;
```

Session 1:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 2;
```

Deadlock

- Атомарность транзакций
- Соблюдайте порядок обновления

ШАРИНГ РЕСУРСОВ

Session 1:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 1;
```

Session 2:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 1;
UPDATE accounts SET balance = balance -
100.00 WHERE acctnum = 2;
```

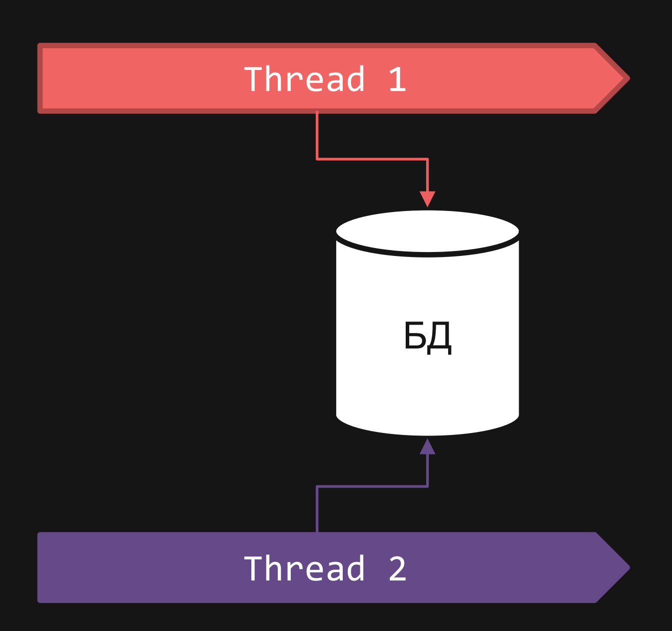
Session 1:

```
UPDATE accounts SET balance = balance +
100.00 WHERE acctnum = 2;
```

Deadlock

- Атомарность транзакций
- Соблюдайте порядок обновления

ШАРИНГ РЕСУРСОВ



Deadlock

- Атомарность транзакций
- Соблюдайте порядок обновления
- Блокирующие запросы – в конец

ДЕМОНЫ



ВЕРНЕМСЯ К ЗАДАЧЕ ПРО ОТЧЕТ

Построить отчет

Набираем сырые данные

Трансформируем

Формируем представление

ВЕРНЕМСЯ К ЗАДАЧЕ ПРО ОТЧЕТ

Набираем сырые данные	Что, если сырые данные приходят к нам по инициативе поставщика?

ВЕРНЕМСЯ К ЗАДАЧЕ ПРО ОТЧЕТ

Набираем сырые данные

Что, если сырые данные приходят к нам по инициативе поставщика?

Мы формируем cron, ждем обращения...

ВЕРНЕМСЯ К ЗАДАЧЕ ПРО ОТЧЕТ

Набираем сырые данные

Что, если сырые данные приходят к нам по инициативе поставщика?

Мы формируем cron, ждем обращения...

Но мы ограничены!

МЫ ХОТИМ СЛУШАТЬ СОБЫТИЯ!



МЫ ХОТИМ СЛУШАТЬ СОБЫТИЯ!



- Непрерывная работа
- Перманентное соединение с источником
- EventDriven-логика
- Возможность ответить на событие по результату

НЕПРЕРЫВНАЯ РАБОТА PHP

```
while (true) {
    // …
}
```

НЕПРЕРЫВНАЯ РАБОТА PHP

```
while (true) {
    // …
}
```

Нет контроля

Процесс никак не отслеживается
Остановка только через halt
Процесс Шрёдингера

PCNTL

- Цикл внутри никуда не денется
- Демон внутри себя знает свой PID

```
$pid = pcntl_fork();
```

- Можно реагировать на сигналы

```
pcntl_signal(SIGTERM, 'my_handler');
```

- Дружба с POSIX

PCNTL



NGINX with FCGI — php ◂ php test.php — 131×55

```
alexanderpryakhin@MBP-Alexander NGINX with FCGI % php test.php
Starting task: fetch_remote_data
Starting task: post_async_updates
Starting task: clear_caches
Starting task: notify_admin
```

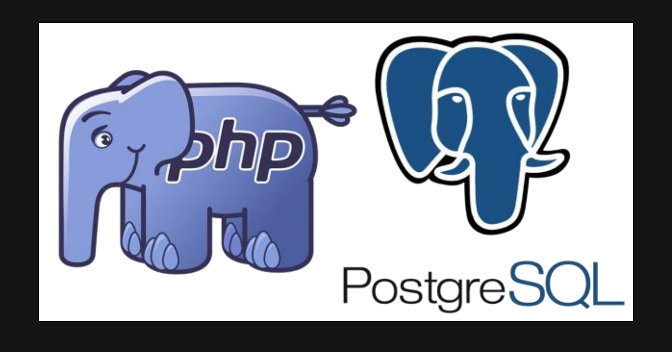
alexanderpryakhin — -zsh — 134×24

```
alexanderpryakhin@MBP-Alexander ~ % ps aux | grep php
alexanderpryakhin 63698   0.0  0.0 408628368   1632 s002  S+    11:40AM   0:00.00 grep php
alexanderpryakhin 63694   0.0  0.0 408715312   2032 s001  S+    11:40AM   0:00.00 php test.php
alexanderpryakhin 63693   0.0  0.0 408715312   1904 s001  S+    11:40AM   0:00.00 php test.php
alexanderpryakhin 63692   0.0  0.0 408724528   1920 s001  S+    11:40AM   0:00.00 php test.php
alexanderpryakhin 63691   0.0  0.0 408715312   1920 s001  S+    11:40AM   0:00.00 php test.php
alexanderpryakhin 63690   0.0  0.1 408715568  18032 s001  S+    11:40AM   0:00.04 php test.php
alexanderpryakhin@MBP-Alexander ~ %
```

ЧТО МОЖНО СЛУШАТЬ ДЕМОНОМ?

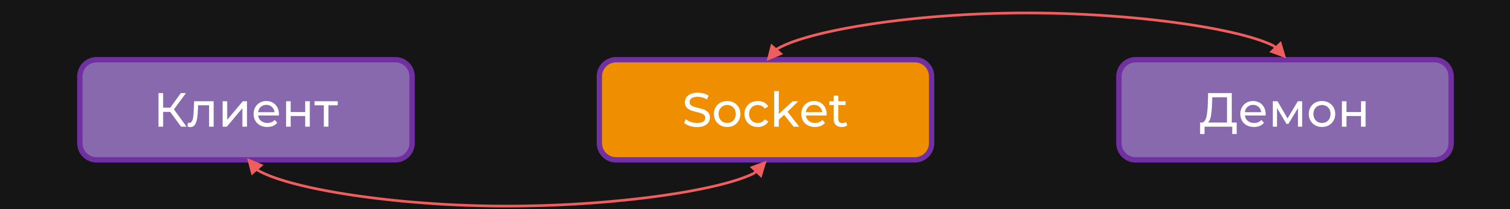
- Постоянный опрос очереди на появление новых событий





ЧТО МОЖНО СЛУШАТЬ ДЕМОНОМ?

- Постоянный опрос очереди на появление новых событий
- Прослушивание Unix- или TCP-сокета



КАК СЛУШАТЬ?

Модуль Event

Можно написать все, вплоть до собственного HTTP-клиента



Модуль Socket

Работает с TCP- и Unix-сокетами



ОСОБЕННОСТИ ПОСТОЯННОЙ РАБОТЫ

Zval-контейнер живёт в памяти, пока на него есть ссылки

ОСОБЕННОСТИ ПОСТОЯННОЙ РАБОТЫ

Zval-контейнер живёт в памяти, пока на него есть ссылки

```
private array $eventData = [];

// …

while(…) {
    $eventData[] = $this->addData();
}
```

Особенности постоянной работы

Zval-контейнер живёт в памяти, пока на него есть ссылки

```
private array $eventData = [];

// …

while(…) {
    $eventData[] = $this->addData();
}
```

Накопление данных со временем работы будет давать утечку памяти.

РАБОТА С ПАМЯТЬЮ

Старый добрый unset

Итераторы и генераторы

```
foreach($iterable as $item){
    $result[] = $item->doSmth();
}
```

```
foreach($iterable as $item){
    yield $item->doSmth();
}
```

ОСОБЕННОСТИ ПОСТОЯННОЙ РАБОТЫ

PHP держит открытым одно соединение к базе на пользователя в процессе.

ОСОБЕННОСТИ ПОСТОЯННОЙ РАБОТЫ

PHP держит открытым одно соединение к базе на пользователя в процессе.

Демон будет работать часами, днями и даже больше.

Соединение может залипнуть.

ЗАЛИПАНИЕ СОЕДИНЕНИЙ

Соединение с ресурсом может залипнуть.

Persistent connect не поможет

В PHP соединение будет висеть, пока не остановится процесс

ЗАЛИПАНИЕ СОЕДИНЕНИЙ

Соединение с ресурсом может залипнуть.

Persistent connect не поможет

Как-то надо мониторить возможность соединения

Процесс один и работает долго. Не получится быстро отследить отвал.

ЗАЛИПАНИЕ СОЕДИНЕНИЙ

Соединение с ресурсом может залипнуть.

Регулярный рестарт демона	Решение в лоб, не гарантирует, что залипаний не случится

ЗАЛИПАНИЕ СОЕДИНЕНИЙ

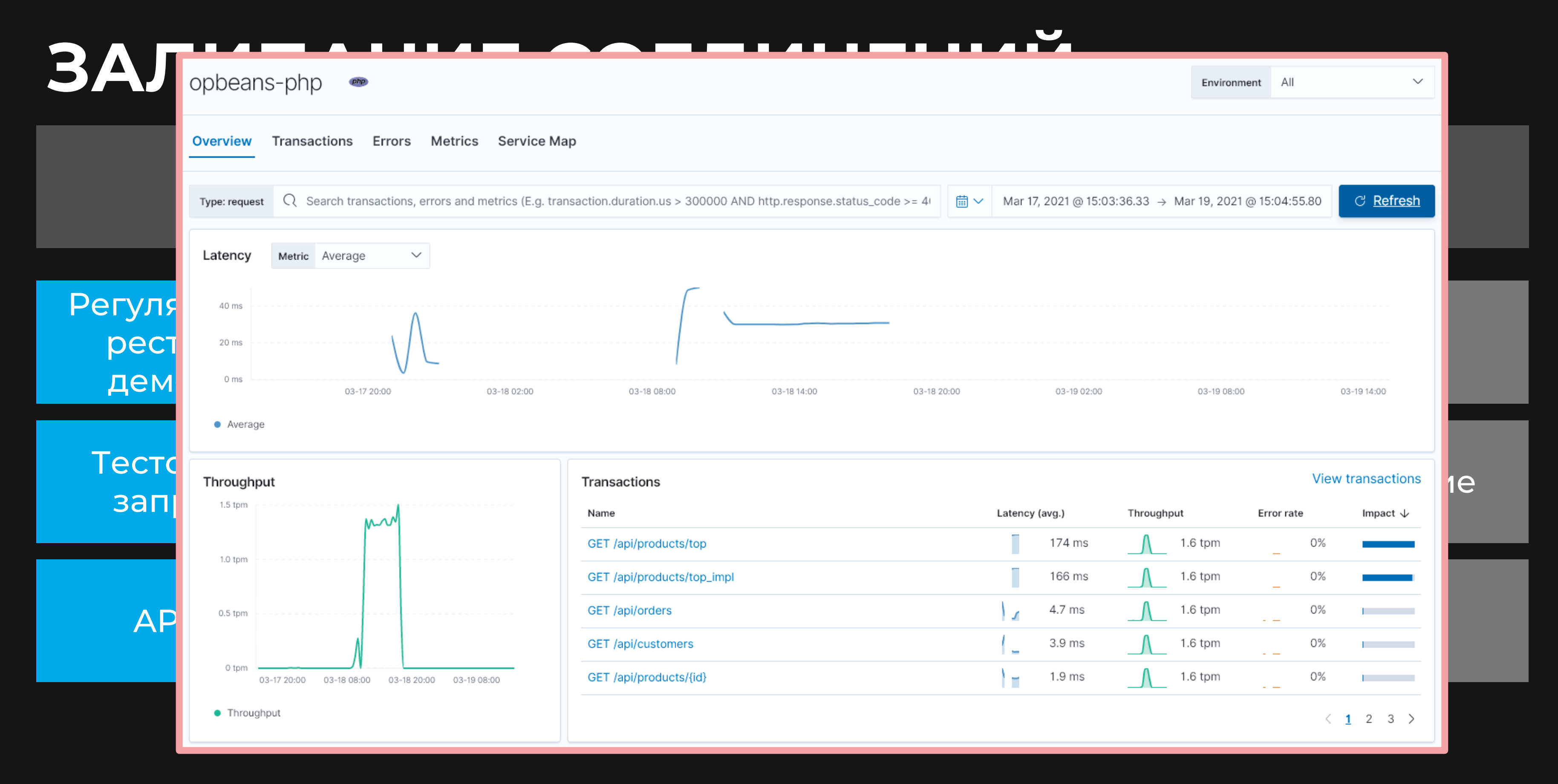
Соединение с ресурсом может залипнуть.

Регулярный рестарт демона	Решение в лоб, не гарантирует, что залипаний не случится
Тестовый запрос	Потребует ручек проверки, но уже помогает проверять состояние

ЗАЛИПАНИЕ СОЕДИНЕНИЙ

Соединение с ресурсом может залипнуть.

Регулярный рестарт демона	Решение в лоб, не гарантирует, что залипаний не случится
Тестовый запрос	Потребует ручек проверки, но уже помогает проверять состояние
APM	ElasticAPM, NewRelic, Pinba – всё, что умеет дёшево и быстро сохранять трейсы и анализировать их


ЗАЛ
Регуля
рест
дем
Тесто
запр
AP
ие
opbeans-php
php
Environment
All
Overview
Transactions
Errors
Metrics
Service Map
Type: request
Search transactions, errors and metrics (E.g. transaction.duration.us > 300000 AND http.response.status_code >= 4
Mar 17, 2021 @ 15:03:36.33 → Mar 19, 2021 @ 15:04:55.80
Refresh
Latency
Metric
Average
40 ms
20 ms
0 ms
03-17 20:00
03-18 02:00
03-18 08:00
03-18 14:00
03-18 20:00
03-19 02:00
03-19 08:00
03-19 14:00
Average
Throughput
1.5 tpm
1.0 tpm
0.5 tpm
0 tpm
03-17 20:00
03-18 08:00
03-18 20:00
03-19 08:00
Throughput
Transactions
View transactions
Name
Latency (avg.)
Throughput
Error rate
Impact
GET /api/products/top
174 ms
1.6 tpm
0%
GET /api/products/top_impl
166 ms
1.6 tpm
0%
GET /api/orders
4.7 ms
1.6 tpm
0%
GET /api/customers
3.9 ms
1.6 tpm
0%
GET /api/products/{id}
1.9 ms
1.6 tpm
0%
1
2
3

СОСТОЯНИЕ ДЕМОНА

Демон может зависнуть, упасть, ~~взорваться~~ из-за критических изменений окружения.

СОСТОЯНИЕ ДЕМОНА

Демон может зависнуть, упасть, ~~взорваться~~ из-за критических изменений окружения.

Нужно отслеживать состояние демона и уметь его переподнимать.

КОНТРОЛЛЕР ДЕМОНОВ

Демон 1

Точка запуска

Демон 2

Демон 3

Мы создаем POSIX-совместимого демона. А значит, знаем его ID в любой момент времени.

Можно хранить соответствие ID и типа демонов, чтобы контролировать их работу.

КОНТРОЛЛЕР ДЕМОНОВ

Точка запуска

Демон 1

Демон 2

Демон 3

Мы создаем POSIX-совместимого демона. А значит, знаем его ID в любой момент времени.

Можно хранить соответствие ID и типа демонов, чтобы контролировать их работу.

Каждому демону пригодится healthcheck-ручка, чтобы проверять жизнеспособность.

КОНТРОЛЛЕР ДЕМОНОВ

Демон 1

Точка запуска

Демон 2

АРМ

Демон 3

Мы создаем POSIX-совместимого демона. А значит, знаем его ID в любой момент времени.

Можно хранить соответствие ID и типа демонов, чтобы контролировать их работу.

Каждому демону пригодится healthcheck-ручка, чтобы проверять жизнеспособность.

КЭШИРОВАНИЕ DNS

DNS кэшируется на момент запуска.

Обращение по URL в случае изменения фактического IP перестанет работать.



ДЕПЛОЙ

Демоны требуют атомарности.

Множество зависимостей усугубит потребление ресурсов и усложнит деплой.

1 демон = 1 задача

Минимум зависимостей

ДЕПЛОЙ

Этап	Описание
Сборка версии	Собираем, тестируем, упаковываем
Сохранение контекста	Старая версия демона должна завершить текущую обработку и не допустить повторной обработки.
Переключение	Демон терпит переключения
Перезапуск	Запускаем новую версию

ИТОГО

PHP жив

Необязательно расширять стек, если нужно организовать большую обработку или ускорить что-то

Не где, а когда!

Разрабатывая демоны, следует думать на более далеком горизонте работы, чем несколько секунд.

Side-эффекты

Не стоит забывать о мониторинге, кэшировании и расшаривании ресурсов

Обратная связь
и комментарии по докладу
по ссылке





PHP Russia
2022

МАТЕРИАЛЫ


PHP создан, чтобы умирать


Async, Swoole, Parallel


Бенчмарк Parallel


PHP EventListener


PHP Sockets


PCNTL example

КОНТАКТЫ


LinkedIn


devenergy.ru


av.priakhin@gmail.com